Author: Bohdan Lukashchuk

Web scraping. Part 1

This part will discuss basic information on web scraping, as well as examples of scraping sites using libraries **Newspaper3k, Requests** and **BeautifulSoup.** Code examples (implemented in Google Colaboratory) with important comments are given below by the link (and QR-code).

Web scraping is a transformation of information from web pages into structured data. It is usually performed using computer programs that download a web page directly via the HTTP protocol, or launch and manage a web browser, like humans do. This section focuses on libraries that download web pages from the server via the HTTP protocol. With them, having only URLs, you can quickly collect a large amount of data from pages with the same structure, for further processing. We will use the **Pandas DataFrame** to store the collected data.

Common areas of use:

- price comparison: some services use web scraping to extract data from online stores and use them to compare product prices;

- collection of e-mails: many companies that use e-mails as a mean of marketing use web scraping to collect them;

- data collection from social media: web scraping is often used to obtain data from social media, such as Twitter, to track global trends in politics, cryptocurrencies, sports, etc. However, most social networks, such as Facebook, are able to track when website manipulations are performed by a 'robot' rather than a real user and block them using captchas or other methods. This can lead to account blocking;

- research and development in various fields: data collection from sites for various types of analysis, research, forecasting, collection and construction of statistical visualization, etc. (eg collection of weather forecasts from various sources, documents, research papers, financial reports, tender results);

- creation of various aggregators of information: news, vacancies, events, etc .;

Web scraping tools

Since each website has its own unique document structure, it is necessary to create a unique web scraper for each case. However, this pays off if you want to get a lot of information from pages with the same structure. It is worth noting that there are paid services that allow you to create web scrapers without writing code, but their functionality is limited.

You must first 'manually' analyze the webpage to identify information location patterns using the developer console. It will inspect the DOM-tree of the page and identify HTML-tags that contain the required information.

The developer console opens in one of the following ways:

- right-click on the website to open the context menu, then select 'Inspect';

- press the key combination: Ctrl+Shift+i/F12/Command+Option+i depending on the operating system and browser;

Then use the information about the page structure and one of the libraries described below to retrieve the data and save it later.

Code examples: shorturl.at/pqLM3

Newspaper3k

Newspaper3k - it is a library created specifically for web scraping of news sites. The way to work with it is as follows:

- Load the page from which you want to collect data by passing an URL to the library method

- use the functionality of the library to analyze and retrieve data from this page

Documentation:
https://newspaper.readthedocs.io/en/latest/

BeautifulSoup + Requests library

Newspaper3k works well for news sites, but errors are still present. If you analyze the column `authors` from the code examples, you can observe that it is often not filled, or filled incorrectly. When you need to parse a non-news site, or have a lot of flexibility when working with the news one, you will need the following libraries.

Requests - HTTP library for Python. It implements various HTTP requests and their processing.

BeautifulSoup is a Python library for extracting data from HTML and XML files. In fact, we will use the **Requests** library to load the HTML page, then parse it with the **BeautifulSoup** library.

Documentation:
https://beautiful-soup-4.readthedocs.io/en/latest/
https://docs.python-requests.org/en/latest/

Requests will not work with SPA web sites (Single Page Application). Because their way of work is to load a 'blank' HTML page, with almost only JavaScript code inside, from the server, then JavaScript code builds the rest of the page. Since we do not get the full HTML page from the server, **BeutifulSoup** cannot parse it. To solve this problem, **Selenium** library can be used, which will be discussed in the next part.

Also the next part will overview **Scrapy** and **lxml**.

Practical tasks:

- using code examples from the links above, write a code to scrap a specific web page. Use the **Newspaper3k** or **BeautifulSoup + Requests** libraries. In case scraper is written in an object-oriented style, the **ScraperFactory** class should be used;

- using scraper from the previous task, collect data from several pages (at least 5) of the same structure and put it to the Pandas DataFrame;

- provide examples of **SPA** pages that cannot be parsed using **BeautifulSoup + Requests** (minimum 1);